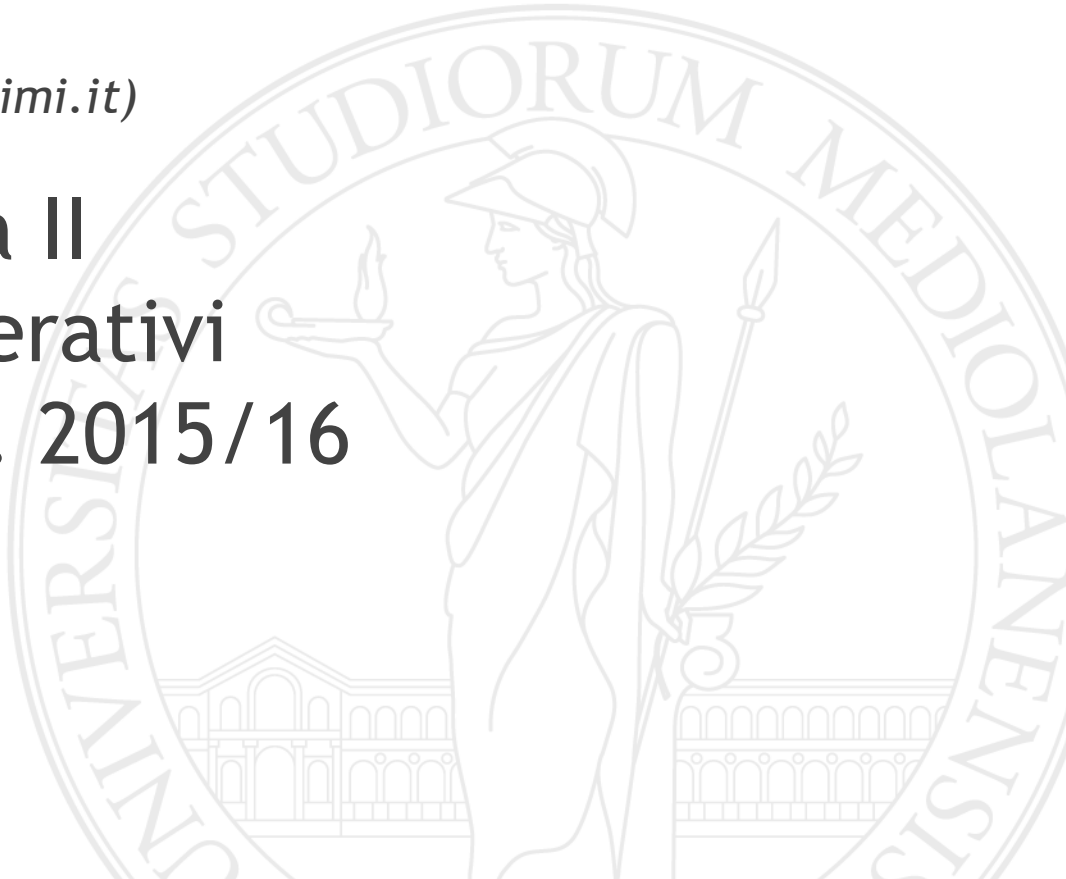
UNIVERSITÀ DEGLI STUDI DI MILANO
DIPARTIMENTO DI INFORMATICA

Alberto Ceselli
(alberto.ceselli@unimi.it)

Informatica II
Sistemi Operativi
DIGIP - a.a. 2015/16



Sistemi Operativi

(modulo di Informatica II)

La memoria virtuale

Patrizia Scandurra

Università degli Studi di Bergamo

a.a. 2014-15

Sommario

- Ambiente virtuale
- Richiesta di paginazione
- Creazione di un processo
  - Copia durante la scrittura
  - File mappati in memoria
- Sostituzione della pagina
- Allocazione dei frame
- Altre considerazioni

Richiami sulla gestione della memoria centrale

- **Allocazione non contigua**:
  - L'immagine di un processo NON è un blocco unico
  - L'immagine di un processo viene spezzata in più parti che sono caricate in memoria in aree non contigue
- Due tecniche fondamentali, spesso combinate:
  - **Paginazione:** l'immagine di un processo è divisa in parti (pagine) di dimensione fissa per un certo SO
    - problemi simili alla gestione a partizioni fisse (frammentazione interna nell'ultima pagina)
  - **Segmentazione:** le parti (segmenti) sono di lunghezza variabile e riflettono la logica del programma (es: dati, istruzioni)
    - problemi simili alla gestione a partizioni variabili (frammentazione esterna)

Memoria virtuale

- Lo spazio di indirizzamento logico può quindi essere più grande dello spazio di indirizzamento fisico
  - *Memoria virtuale* o *ambiente virtuale*
- Il termine "virtuale" indica una cosa che **NON esiste fisicamente ma solo logicamente** (concettualmente, metaforicamente,…)
- **Per i processi è come se l'intera immagine fosse in memoria principale**
  - il SO si occupa di fare in modo che i processi non si accorgano che l'immagine non è interamente in memoria
  - permette agli spazi di indirizzamento di essere condivisi da numerosi processi
  - permette una maggiore efficienza nella crezione del processo

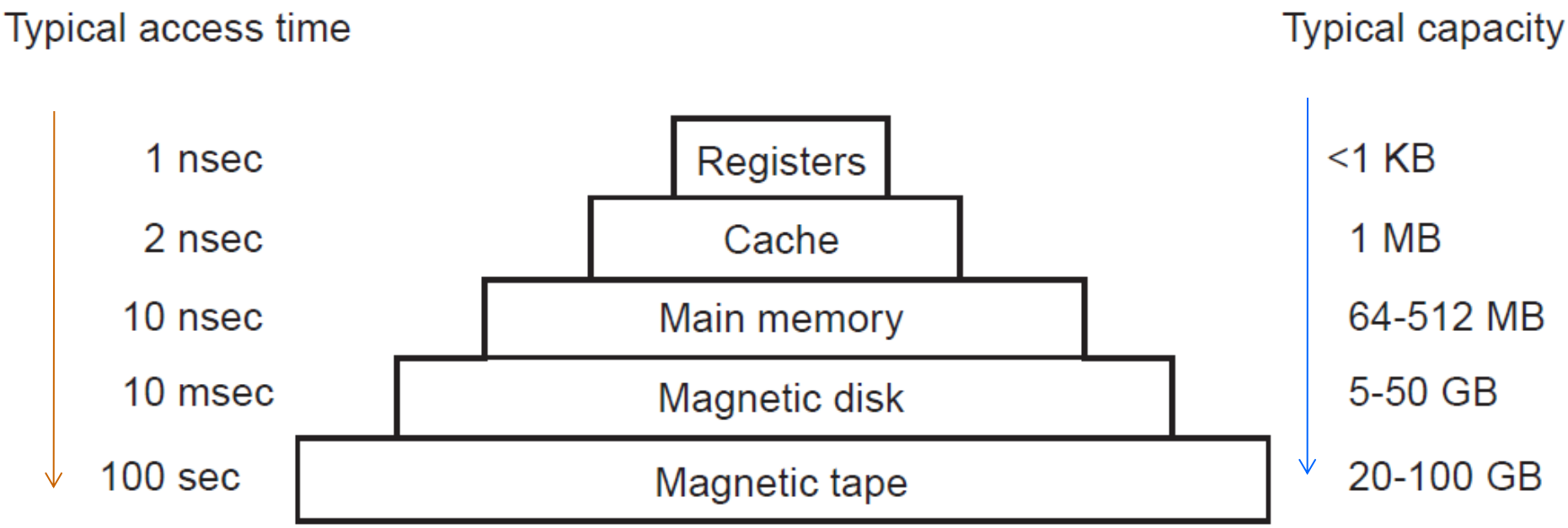
Considerazioni sulla virtualizzazione della memoria centrale

- Considerazione: in una certa fase dell'esecuzione **di un processo solo alcune parti** dell'immagine **vengono realmente usate** (lette o scritte)

- Conseguenze (benefici) fondamentali:
  - In un certo istante **si possono tenere solo le parti necessarie in memoria principale** (le altre sono su memoria secondaria) **=> più processi in memoria**
  - L'immagine di **un processo può superare la dimensione fisica reale della memoria principale**
  - **Per caricare/scaricare un programma utente in/dalla memoria sono necessarie meno operazioni**

Elementi di memoria di un elaboratore

La memoria di un sistema di elaborazione è tipicamente organizzata nella seguente **gerarchia di memorie**



- La capacità del sistema è pari a quella della sola memoria centrale (la cache contiene solo *copie*!)
- Il gestore di memoria può però gestire anche parte della memoria secondaria, al fine di **emulare memoria principale**

Gerarchia di memoria

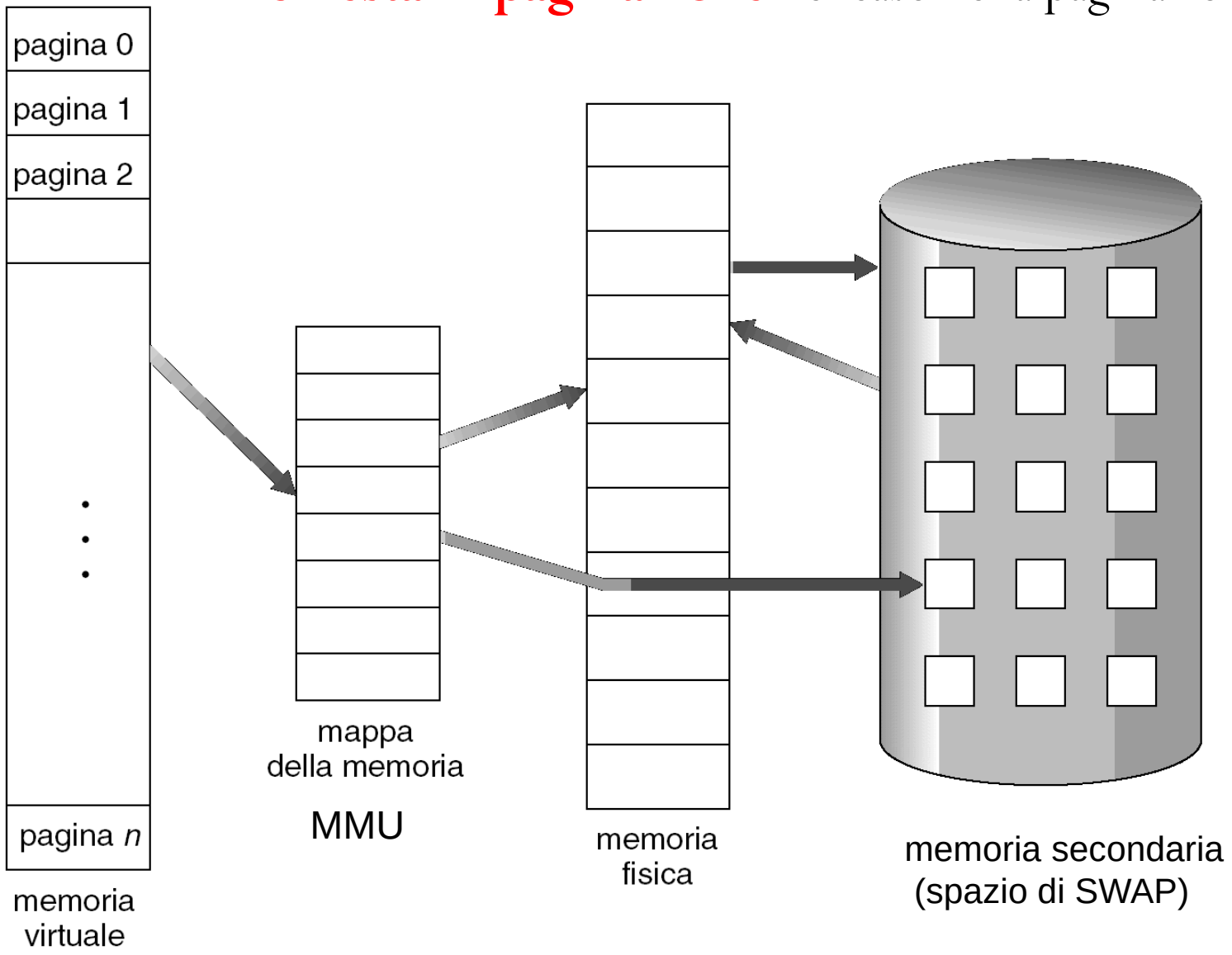
- La memoria viene organizzata in livelli caratterizzati da velocità, dimensioni e costi diversi
- I blocchi di informazione possono essere trasferiti da un livello inferiore a uno superiore
- Cerco di tenere i ***blocchi di informazione usati più di frequente*** **vicino alla CPU**, per ottimizzare i tempi
- **Obiettivo**: **fornire la sensazione di una memoria con la velocità del primo livello e la capacità del (dei) successivo(i)**

Implementazione dell'ambiente virtuale

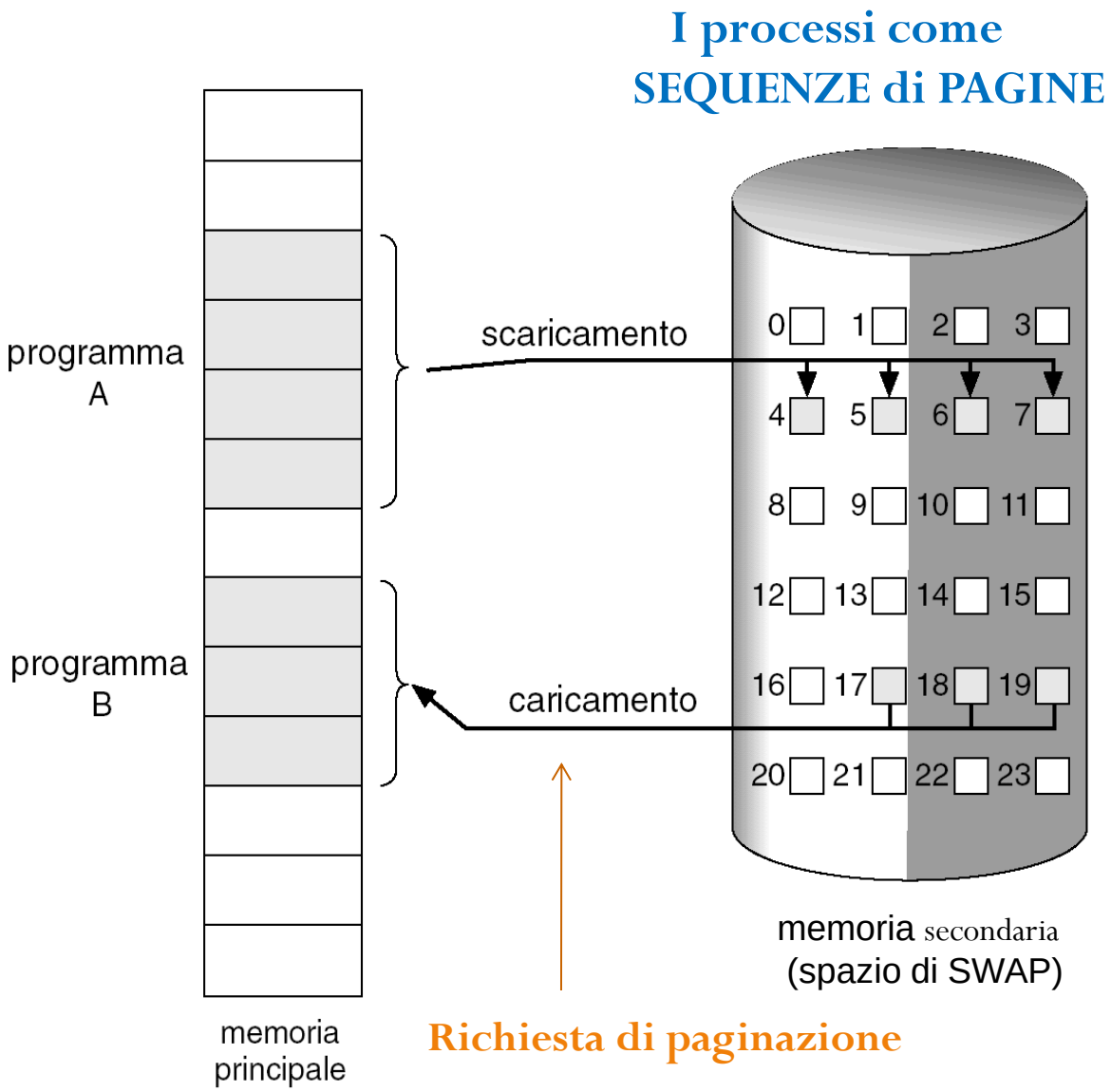
- La memoria virtuale può essere implementata attraverso:
  - **Richiesta di paginazione**
  - **Domanda di segmentazione** *[non in programma]*
    - Più complessi, perchè i segmenti hanno dimensione variabile
    - Parecchi sistemi forniscono uno schema di segmentazione paginata

Memoria virtuale più grande della memoria fisica

Richiesta di paginazione nel caso della paginazione…



Trasferimento di una memoria paginata nello spazio contiguo del disco



Richiesta di paginazione

- **Introdurre una pagina in memoria solo se è necessario**
  - Meno dispositivi di I/O necessari
  - Meno memoria fisica necessaria
  - Risposta più veloce
    - diminuisce il tempo di context switch
  - Più utenti
- Occorre distinguere tra **pagine in memoria e non** in memoria, ed ancora tra **riferimenti legali e illegali**
- Mentre il processo esegue ed accede ad una sua pagina ⇒ riferimento ad essa:
  - **Se il riferimento è illegale ⇒ termine del processo**
  - **Se legale, ma non in memoria ⇒ portare in memoria**

Bit di validità

- Ogni elemento della tabella delle pagine ha un **bit di validità**:
  1 ⇒ valida e in-memoria,
  0 ⇒ non valida o non in memoria
  - Inizialmente, il bit è impostato a 0 per tutti gli elementi
- Durante la traduzione dell'indirizzo, se il bit di validità è 0 ⇒ **mancanza di pagina** (**page fault**)

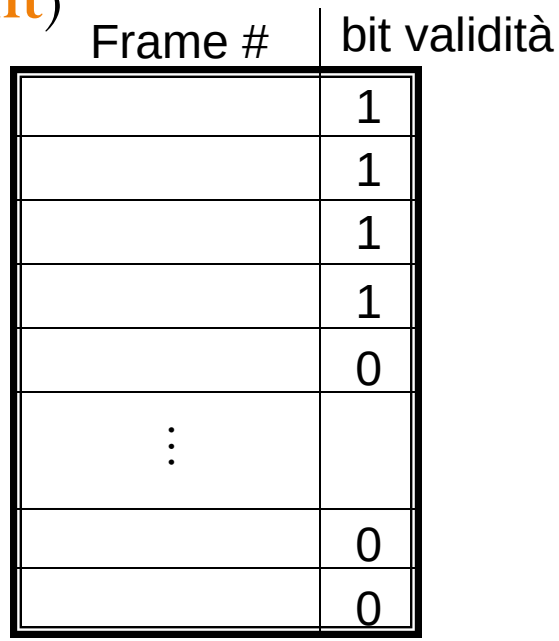
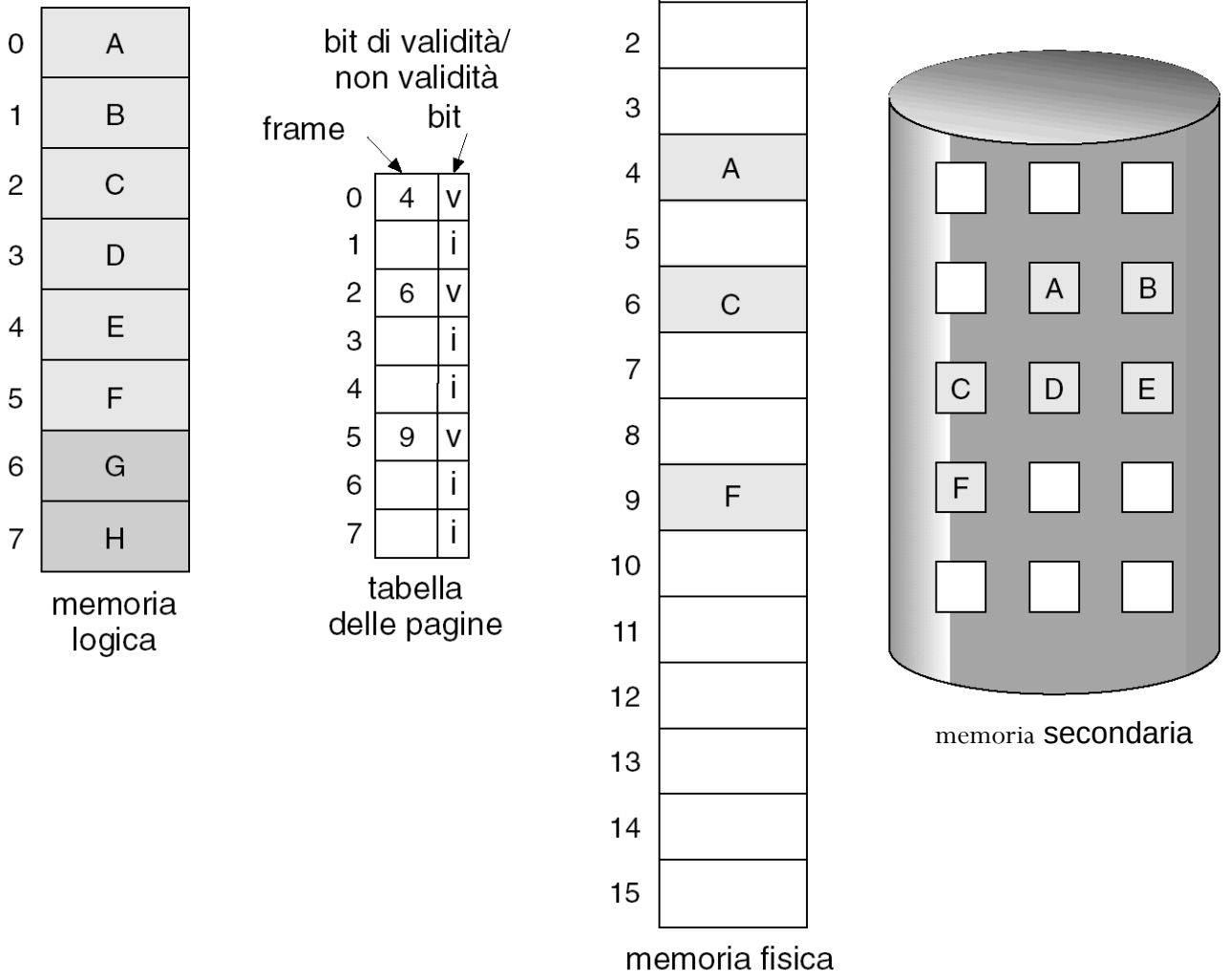
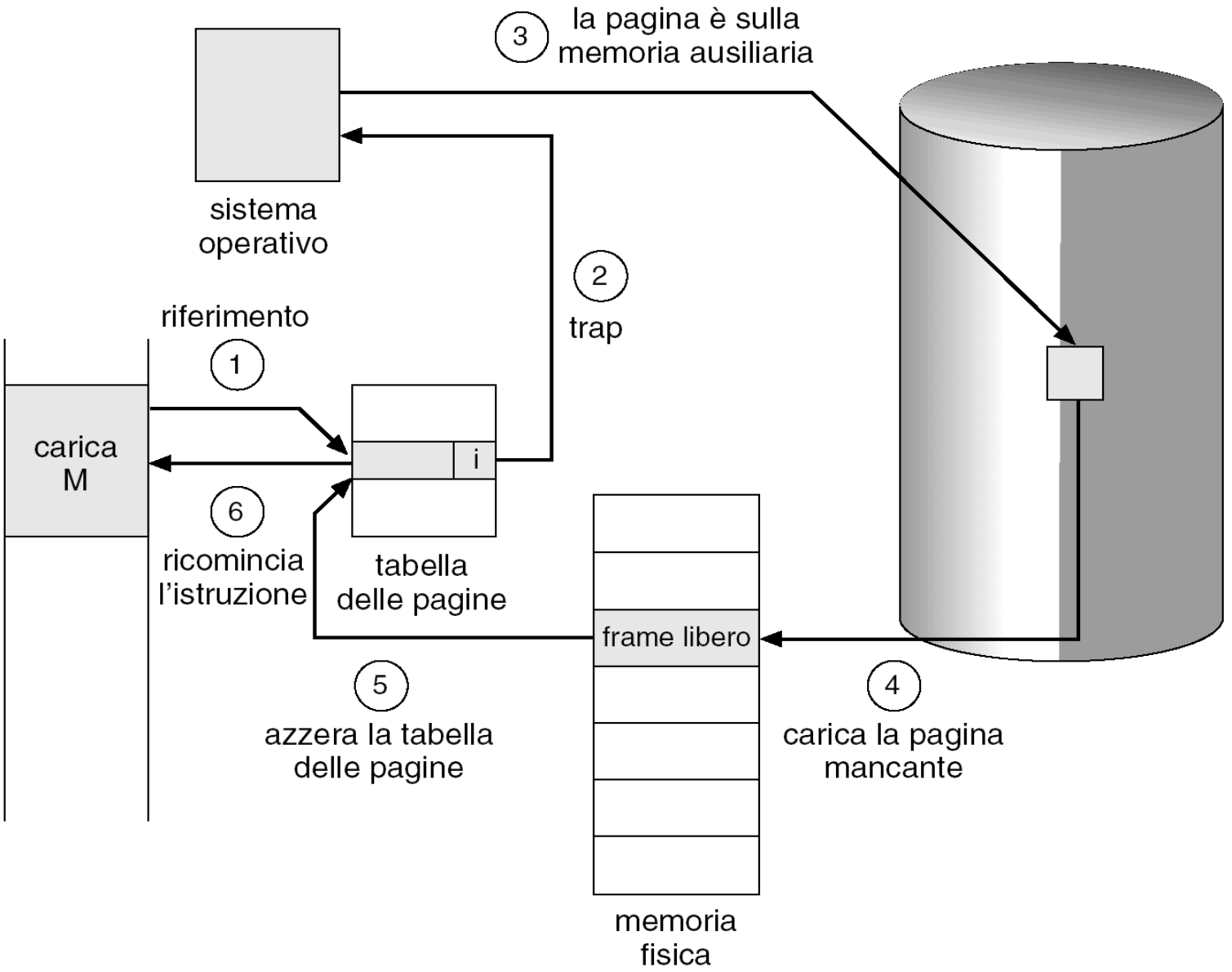


Tabella delle pagine

Tabella delle pagine quando alcune pagine non sono nella memoria centrale

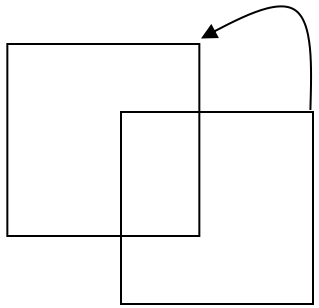


Gestire una mancanza di pagina



Gestire una mancanza di pagina: dettagli del lucido precedente

1. Se il processo accede ad una pagina non valida ⇒ trap di mancanza di pagina (*page fault*) nel SO
2. Esaminare la PT per decidere:
   - Riferimento non valido ⇒ termine del processo
   - Solo non in memoria, vai a 3.
3. Cercare un frame libero
4. Caricare dal disco la pagina in un frame libero
   - esegue un *context switch* (opzionale)
5. Modificare la tabella, validità del bit = 1
6. Fa ripartire l'istruzione interrotta dall'eccezione



Prestazione della richiesta di paginazione

- Probabilità di mancanza di pagina $0 \leq p \leq 1.0$
  - se $p = 0$ non ci sono mancanze di pagina
  - se $p = 1$, ogni riferimento è una mancanza di pagina

- Tempo di accesso effettivo (EAT)

EAT = (1 – *p*) x tempo di accesso alla memoria

+ *p* x tempo di mancanza della pagina

dove:

tempo di mancanza della pagina =

tempo per attivare il servizio di page fault + swap page in + tempo di ripresa del processo

Esempio di richiesta di paginazione

- Tempo di accesso alla memoria = 200 ns
- Tempo medio per il page-fault = 8 ms = 8 000 000 ns
- EAT = (1 – p) x 200 + p x 8 000 000
  = 200 + p x 7 999 800 ns
- EAT è direttamente proporzionale al tasso p di page fault
  - **È importante mantenere p basso** per non avere un degrado delle prestazioni del sistema
  - Se ad es. un accesso su 1000 causa un page fault, allora EAT = 200 + 7999,8 ns = 8199,8 ns : un rallentamento di un fattore 40 (200 ns x 40 = 8000 ns)!

Cosa succede se non ci sono frame liberi?

- **Sostituzione di pagina** – trovare alcune pagine in memoria, ma non veramente in uso, e scambiarle di posto
  - Algoritmo
  - Prestazione – l'algoritmo deve essere tale per cui venga **minimizza**to **il numero di page fault futuri**
- La stessa pagina può essere portata in memoria più volte

Creazione di un processo

La memoria virtuale permette altri benefici durante la creazione del processo, attraverso le tecniche:

- **Copia durante la scrittura (copy-on-write)**
  - simile alla tecnica di condivisione delle pagine
  - riduce la richiesta di paginazione da parte del nuovo processo creato
- **File mappati in memoria**
  - permette l'accesso a *file* su disco tramite il sottosistema di gestione della memoria centrale (più efficiente), anzicchè il sotto-sistema di gestione dell'I/O

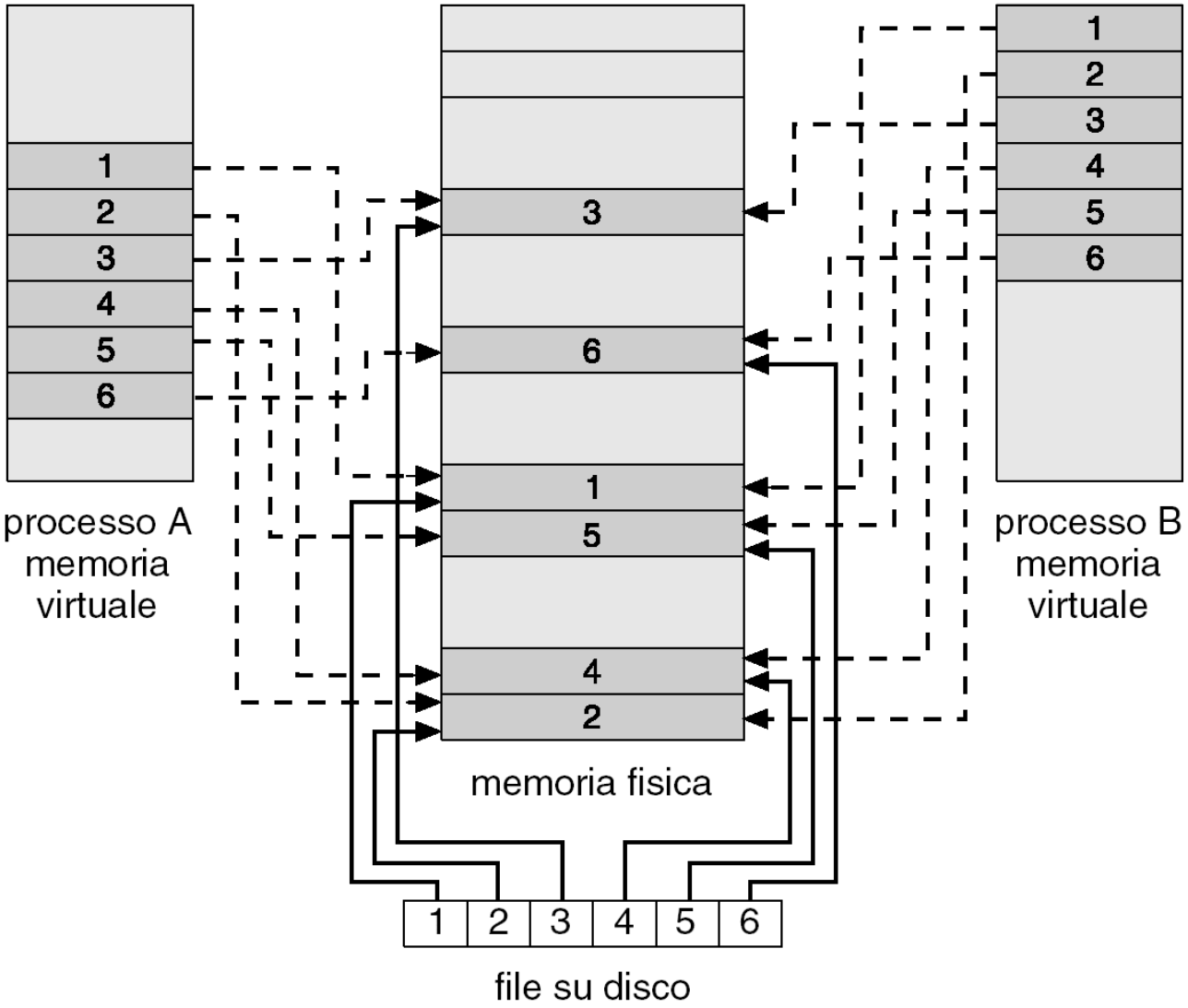
Copia durante la scrittura

- La chiamata fork() crea il processo figlio come un esatto duplicato del padre (copia delle pagine del padre)
- La "copia durante la scrittura" ("copy-on-write") **permette ai processi padre e figlio di *condividere* inizialmente le stesse pagine in memoria**
  - Pagine contrassegnate come "copy-on-write"
- **Se uno dei processi (il padre o il figlio) scrive in una pagina condivisa allora viene creata una "copia" della pagina per il figlio**
  - Potrebbero esserci zero richieste, se il processo figlio esegue subito exec()
  - Le pagine libere sono tipicamnete scelte da ***pool* di pagine *riempi con zero alla richiesta*** (cioè azzerate prima di essere allocate)

File mappati in memoria

- Tecnica che permette ad un file I/O di essere trattato tramite la procedura di gestione della memoria centrale, ***mappando*** **un blocco del disco ad una (o più) pagine in memoria virtuale**
- Alcuni SO forniscono la mappatura dei dati in memoria solo con una chiamata di sistema specifica
  - In Unix, Linux: *mmap()*
  - Win32: *createFileMapping()*
- L'accesso iniziale al file (*open()*) procede con **la richiesta di paginazione**
  - Una porzione del file delle dimensioni di una pagina viene letta dal file system e caricata in una pagina fisica
- Le successive **letture/scritture del file** sono trattate **come accessi alla memoria centrale**
- Alcuni SO, come Solaris, usano solo file mappati: nella memoria del processo o del kernel, a seconda della chiamata di sistema usata

File mappati in memoria: condivisione di file



I processi possono mappare lo stesso file in modo concorrente (e sincronizzato!), per consentire la condivisione dei dati

Mappature in memoria dell'I/O

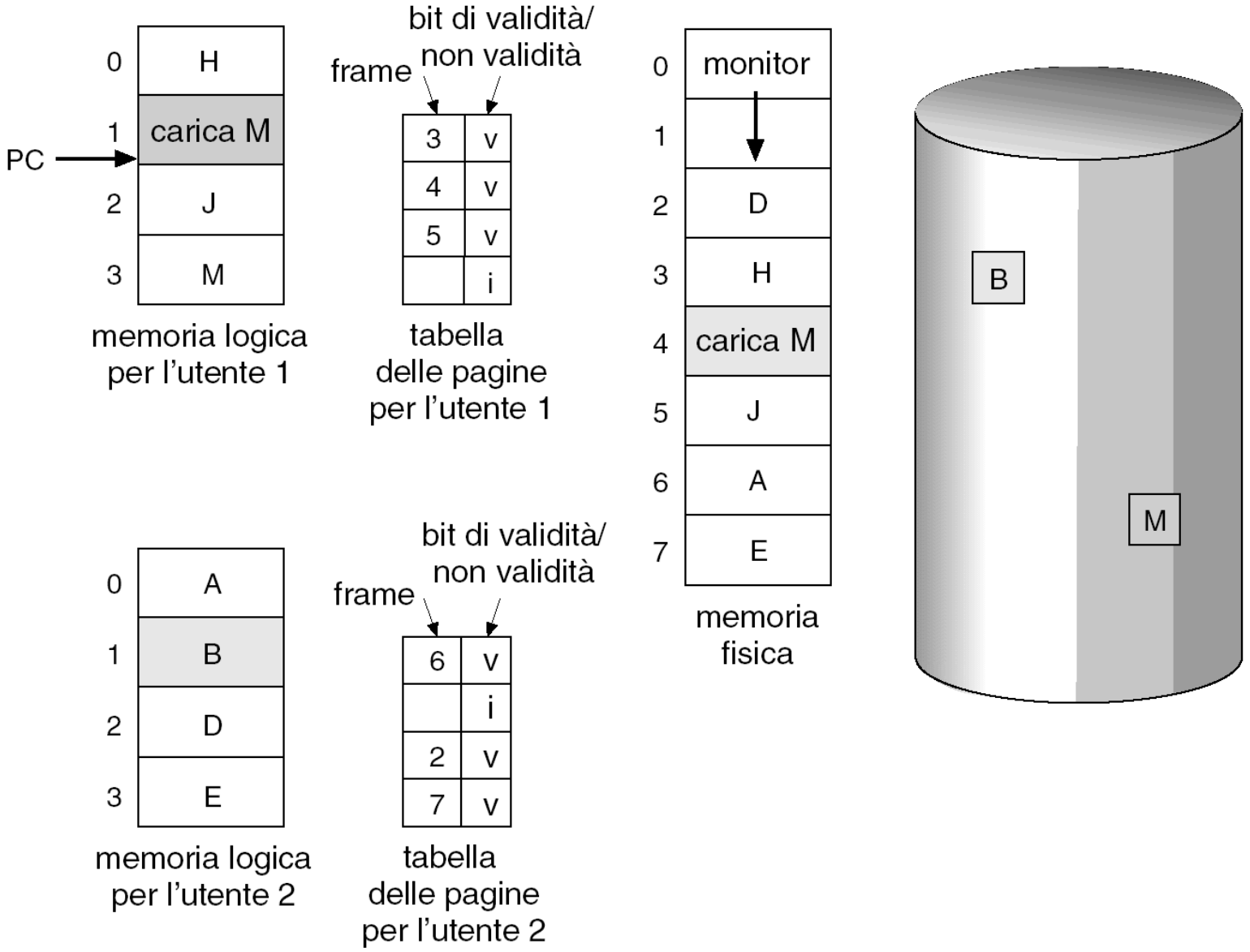
- Convenzionalmente, esistono istruzioni dedicate per il trasferimento dei dati di I/O dai registri del controllore della periferica ai e la memoria e viceversa
- **Mappatura in memoria dell'I/O**: tecnica adottata da molte architetture degli elaboratori per rendere più agevole l'accesso ai dispositivi di I/O (controllori video, porte seriali e parallele per modem e stampanti, ecc..)

- Indirizzi di memoria sono riservati per la mappatura dei registri dei dispositivi

Sostituzione delle pagine

- La memoria fisica è solitamente molto più piccola della memoria logica dei programmi presenti nel sistema
- **Ad un certo punto, ci sarà un *page fault* per il quale non ci saranno frame liberi nel sistema**
- **Soluzione**: sostituire pagine possibilmente quelle "*che serviranno meno in futuro*"…
  - É conveniente rimpiazzare le pagine in sola lettura o non modificate (con *bit di modifica* a zero)
    - Perché non è necessario copiarle nell'area di swap
  - Diverse *politiche di scelta* di tali pagine

Necessità di sostituzione della pagina



La richiesta di paginazione

Per implementare la richiesta di paginazione occorre:

1. Un **algoritmo di allocazione dei frame**
   - per decidere quanti frame assegnare ad un processo

2. Un **algoritmo di sostituzione della pagina**
   - per selezionare i frame da sostituire, solo se dal punto 1. emerge che la memoria è satura

Richiesta di paginazione: procedura base

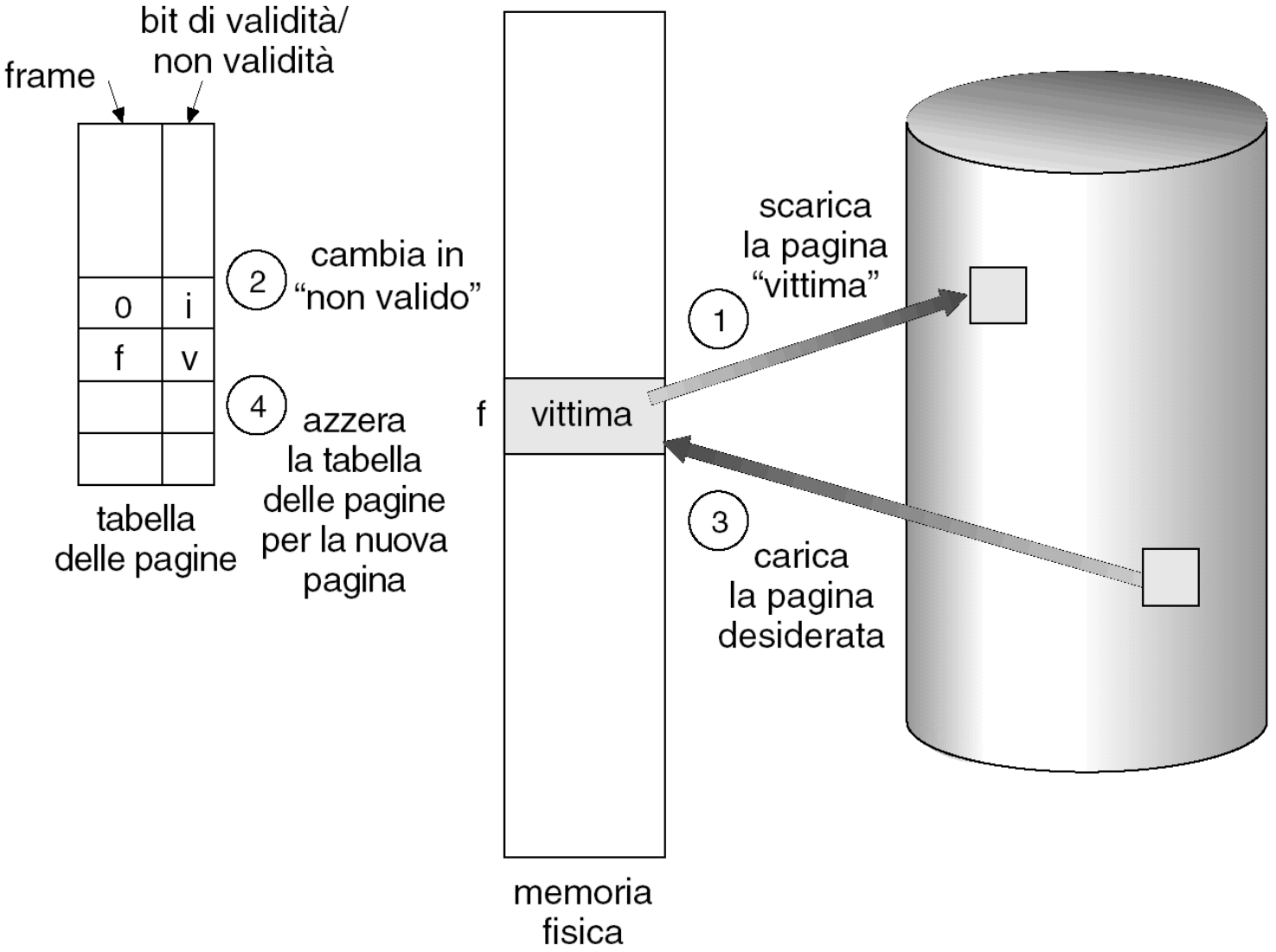
1. **Trovare la** posizione della **pagina** desiderata **sul disco**

2. **Trovare un frame libero**:
   a) Se c'è un frame libero, usarlo
   b) Altrimenti, usare **l'algoritmo di sostituzione della pagina** per selezionare un ***frame vittima***

3. **Leggere la pagina desiderata nel frame libero**; **cambiare le tabelle** dei frame e delle pagine

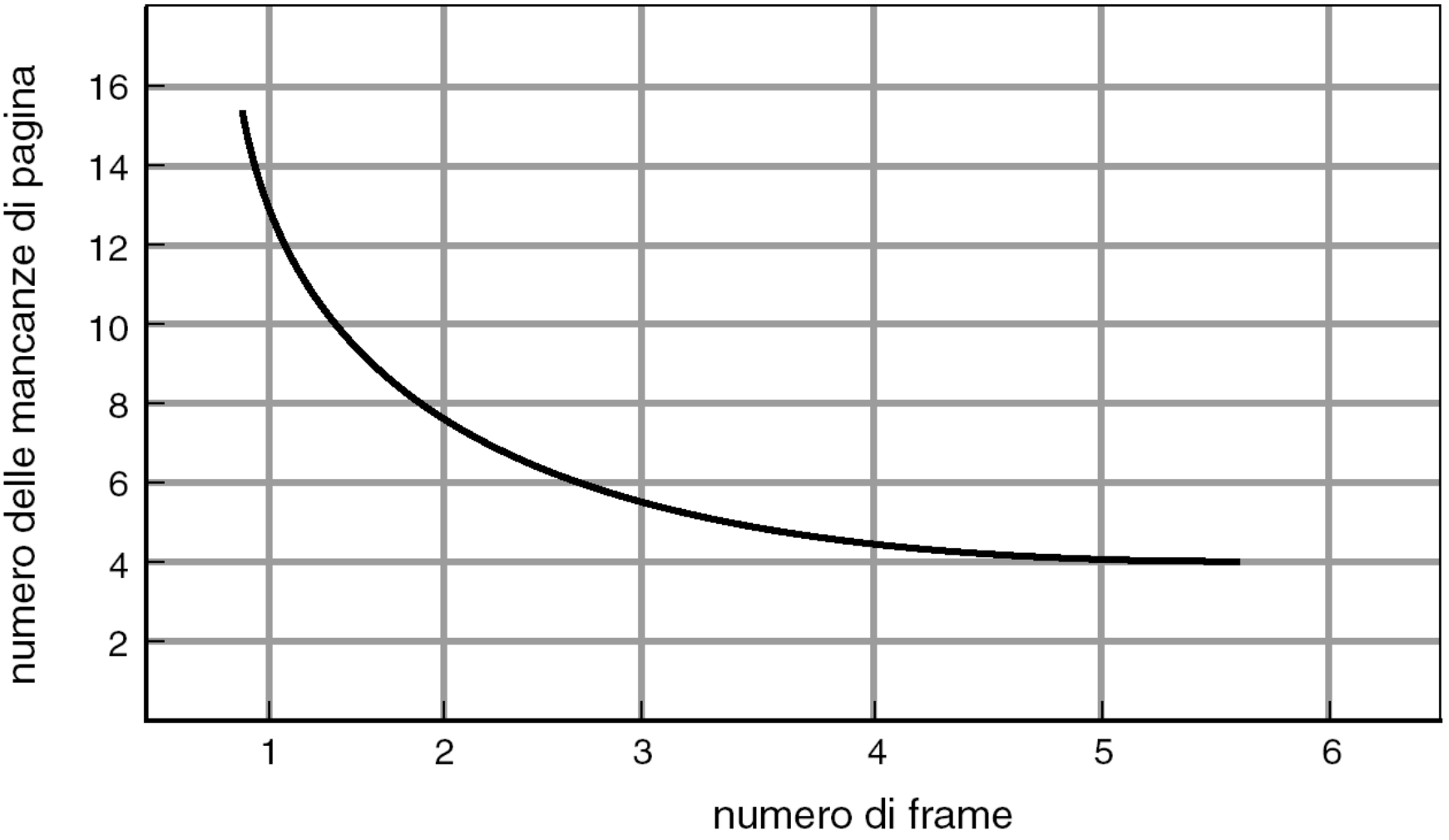
4. **Riprendere il processo**

Passi per la sostituzione della pagina



1. e 3. comporta il trasferimento di due pagine: raddoppia il tempo di servizio della mancanza di pagina e aumenta il tempo EAT!

Grafico delle mancanze di pagina in funzione del numero di frame



Algoritmo di sostituzione della pagina

- **Goal**: desiderare il più basso tasso di mancanza di pagine
- **Un algoritmo viene "valutato"** facendolo operare
  - su esempi di "sequenze di richieste di pagine" (da campioni reali o random) – *stringa di riferimento*
  - calcolando il numero di page fault

- In tutti i nostri esempi la stringa di riferimento sarà:

  1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

Politiche di rimpiazzamento delle pagine (1)

- ***First In First Out***
  - Scarica le pagine nel sistema da più tempo
    - Timer associato ad ogni pagina
  - Si rischia di eliminare pagine molto utili

- ***FIFO Second Chance***
  - Si applica una politica FIFO
    - Se il bit *usata è 0 la pagina viene scaricata*
    - Se è 1 tale bit e il timer vengono azzerati e si continua a cercare
    - Alla successiva scansione, se il bit *usata è ancora a zero* la pagina viene scaricata

Politiche di rimpiazzamento delle pagine (2)

- ***Least Recently Used (LRU)***
  - Scarica le pagine non usate da più tempo
  - Ad ogni pagina è associato un timer di "*vecchiaia*"
- ***Not Recently Used (NRU)***
  - Scarica le pagine non usate di recente
  - Si basa sui bit *usata e modificata*

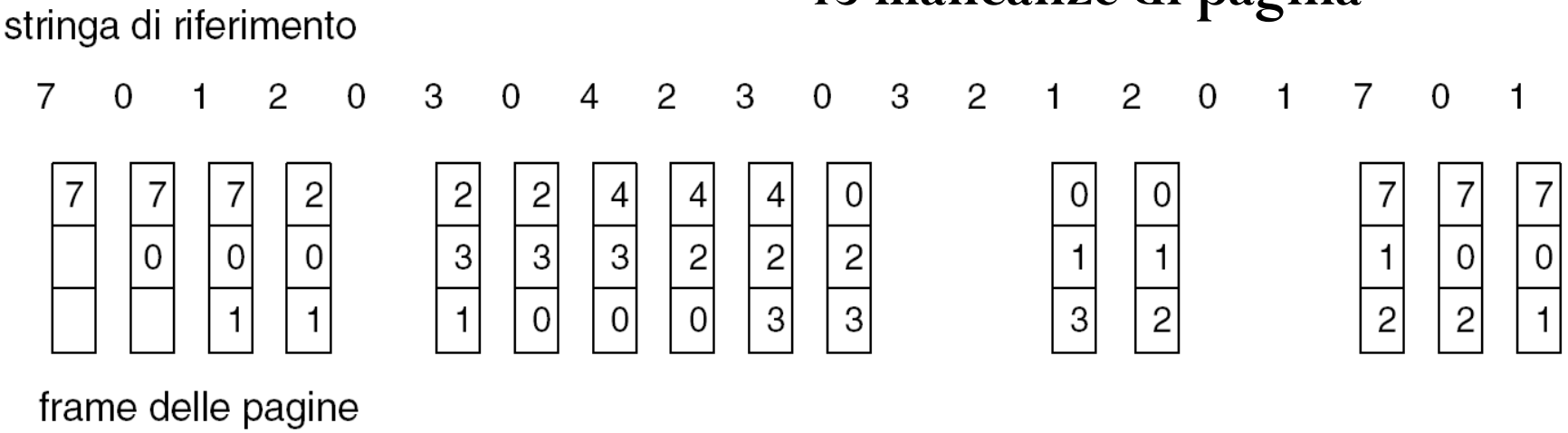
Sostituzione FIFO della pagina

Algoritmo più semplice, implementa **la politica:**
scegli la pagina in assoluto più vecchia!

- Timer associato ad ogni pagina per il tempo di caricamento, o
- Una coda: sostituzione in cima, caricamento in coda

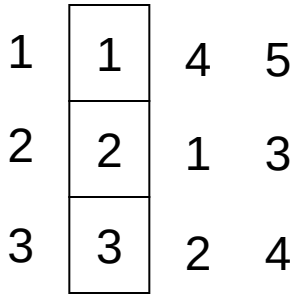
Esempio: **3 frame** (in memoria possono esserci per processo 3 pagine allo stesso tempo)

15 mancanze di pagina



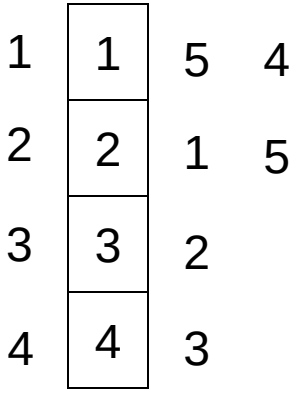
Sostituzione FIFO della pagina (cont.)

- Stringa di riferimento: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5
- **3 frame**



9 mancanze di pagina

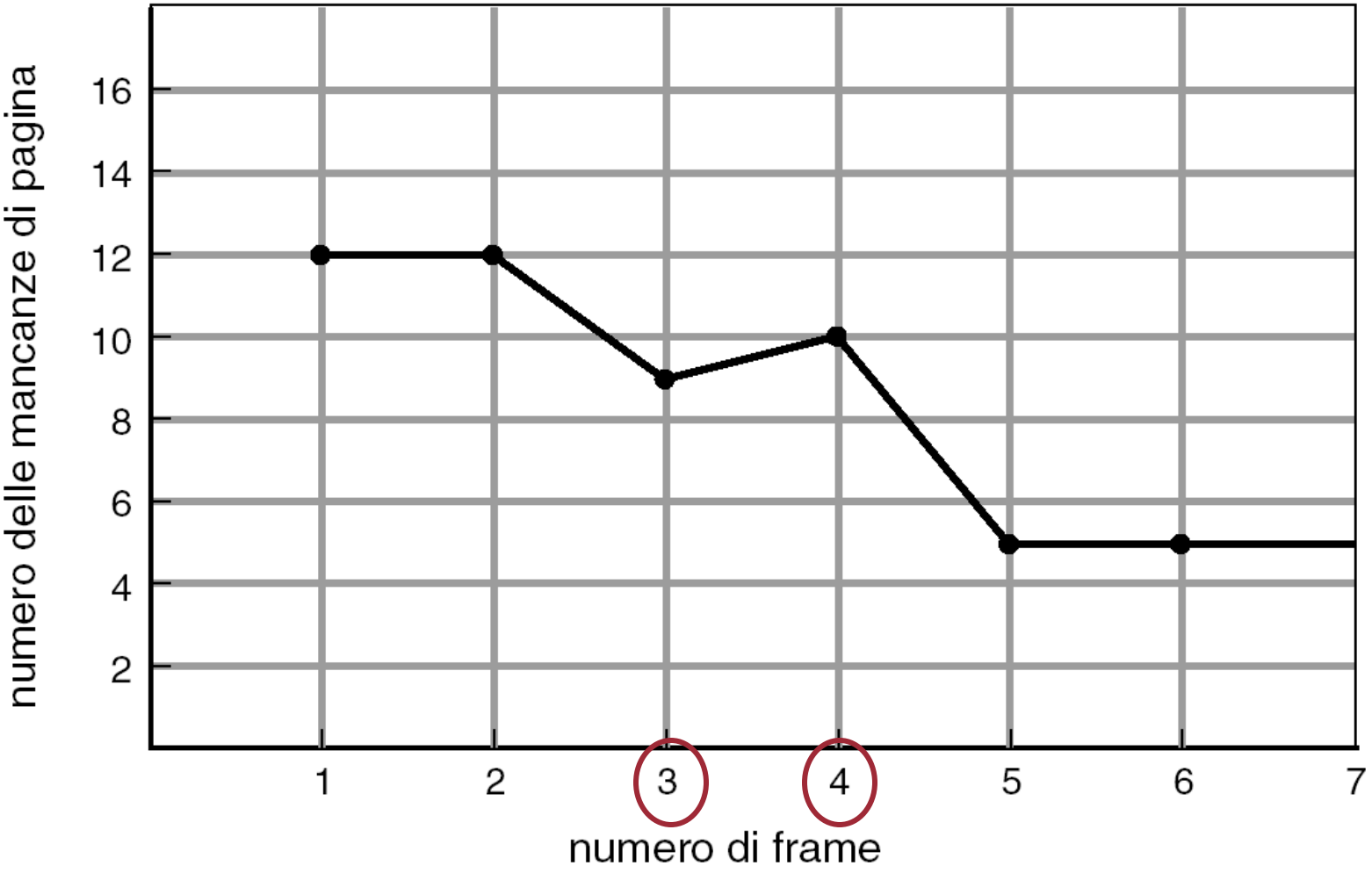
- **4 frame**



10 mancanze di pagina :-O

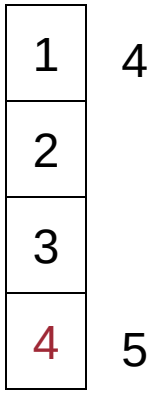
- Sostituzione FIFO – **Anomalia di Belady**
  - più frame $\Rightarrow$ più mancanze di pagina!

Sostituzione FIFO che illustra l'anomalia di Belady



Algoritmo utopia

- Non soffre dell' anomalia di Belady
- Produce il numero minimo di mancanze di pagina
- **Pagina vittima**: Sostituire la pagina **che non sarà usata per il più lungo periodo di tempo**
- **4 frame**: 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

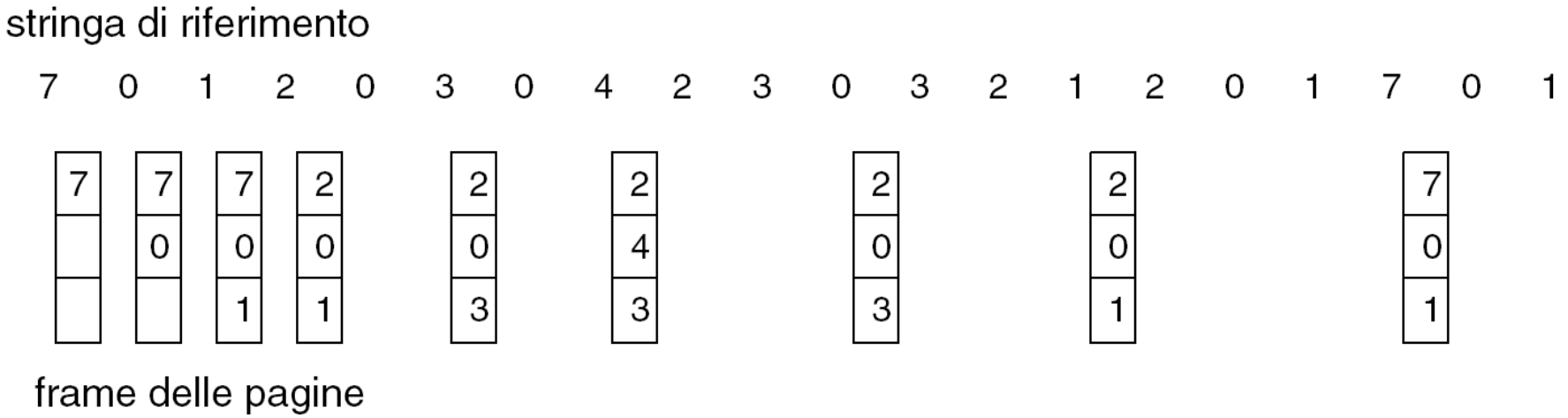


Solo **6 mancanze di pagina**!

- Come sapete questo? Algoritmo esatto, ma **solo IDEALE**
- Usato per misurare quanto sono buone le prestazioni dell'algoritmo

Sostituzione ottimale della pagina

Ancora un esempio:



9 mancanze di pagina ⇒ nessun algoritmo "vero"

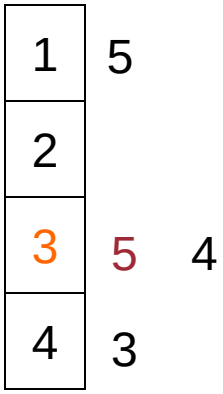
può elaborare questa stringa a 3 frame con meno di 9 pagine

Algoritmo LRU (Least-Recently-used)

- **Pagina vittima**: Sostituire la pagina **che non è stata usata per il periodo di tempo più lungo**

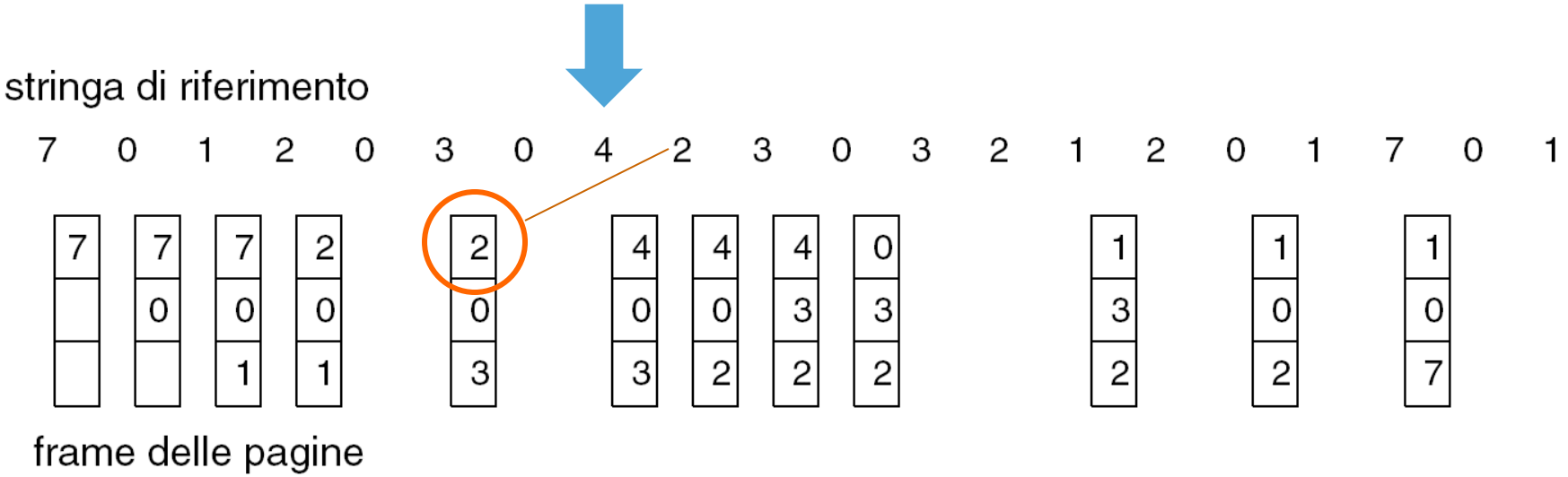
- Stringa di riferimento:
  1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

  Pagine vittime selezionate nell'ordine: 3,4,3,1



Sostituzione LRU della pagina

Ancora un esempio:



12 mancanze di pagina ⇒ meglio di FIFO (15)

Implementazione dell'algoritmo LRU

- Il problema è determinare un ordine dei frame, dal momento del "loro ultimo uso"
- Può richiedere supporto hardware
- Due possibili implementazioni:
  - tramite **contatore di vecchiaia**
  - tramite **stack**

Non mostrano mai l'anomalia di Belady

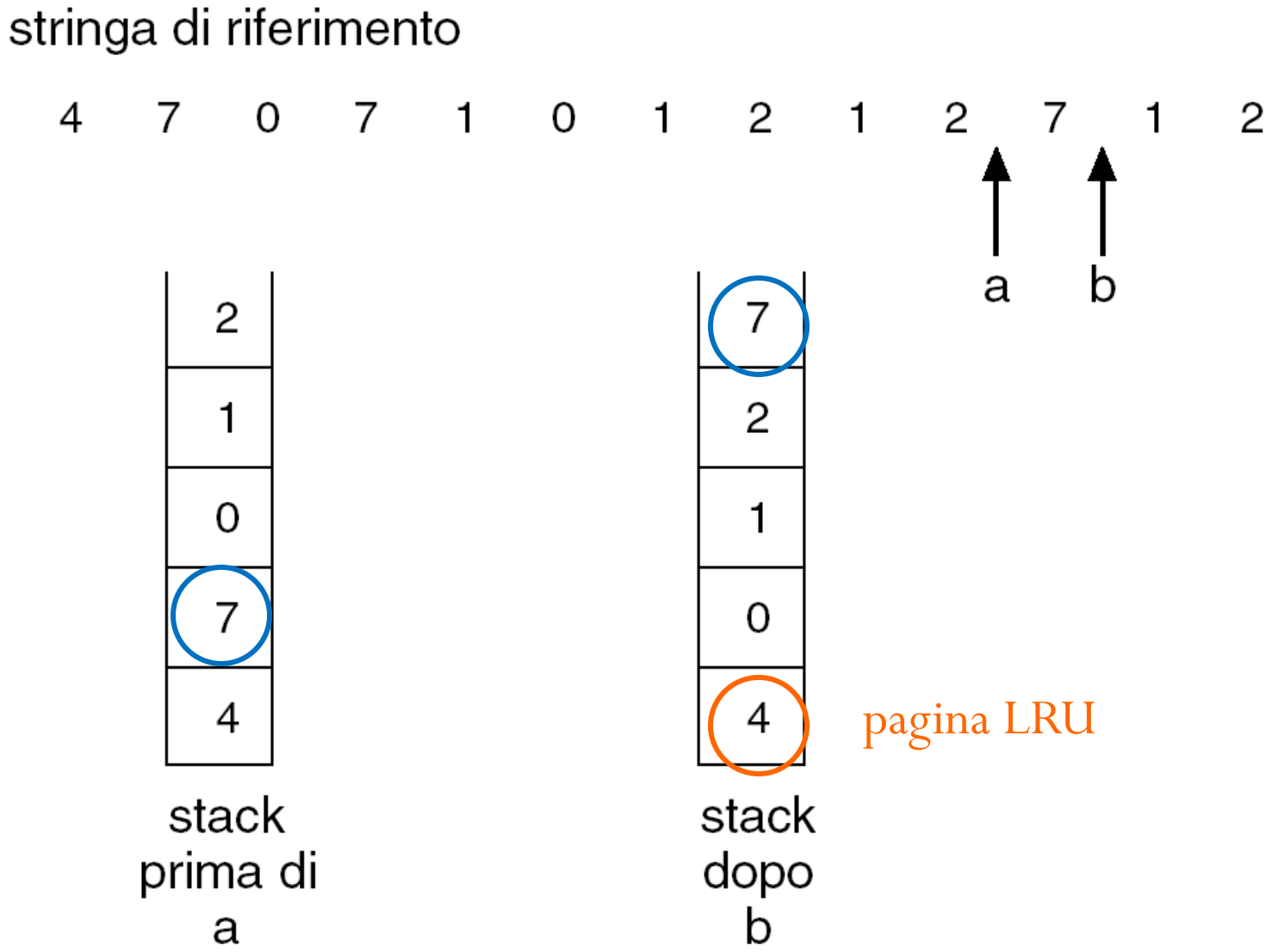
Implementazione dell'algoritmo LRU – Contatore di vecchiaia

- Si aggiunge alla **CPU un orologio o un contatore logico**
- **Ogni pagina ha un contatore**: ogni elemento della tabella delle pagine ha un campo per il tempo d'uso
- **Accesso alla memoria**: ad ogni riferimento ad una pagina, il contenuto del registro dell'orologio "viene copiato" nel campo del tempo d'uso della tabella delle pagine per quella pagina
- **Sostituzione della pagina**:
  - Pagina LRU: la pagina con il più piccolo valore del contatore
  - Richiede una ricerca nella PT + una scrittura in memoria per ogni accesso alla memoria
  - Overflow dell'orologio

Implementazione dell'algoritmo LRU - Stack

- Adatto alle implementazioni puramente software
- Mantenere uno stack dei numeri di pagina: una lista a doppio collegamento con puntatore all'elemento iniziale e finale
  - **Riferimento ad una pagina**:
    - mettere la pagina in cima allo stack
    - richiede di cambiare al più 6 puntatori (per l'estrazione in mezzo)
  - **Nessuna ricerca per la sostituzione** ☺
    - Pagina LRU: la pagina in fondo allo stack

Uso di uno stack per registrare i riferimenti alle pagine usate più di recente



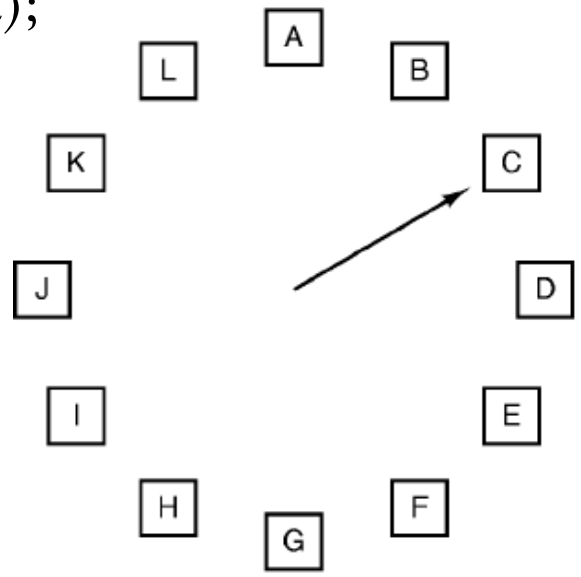
Approssimazione dell'algoritmo LRU (1)

- In verità, entrambe le implementazioni (contatori, stack) appesantiscono il sistema
  - perché l'aggiornamento dei campi orologio o dello stack deve essere eseguito per ogni riferimento alla memoria!

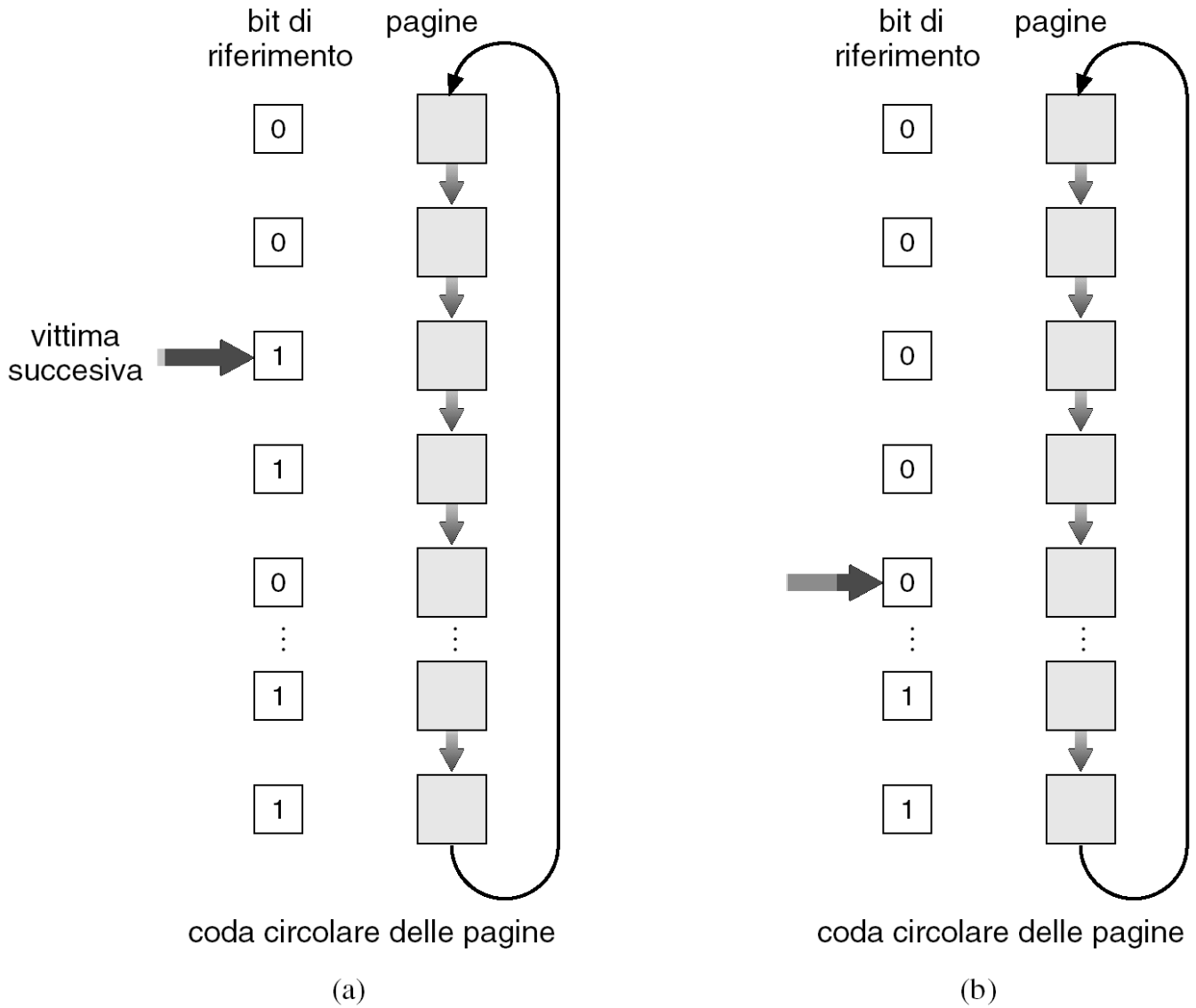
- Molti computer, forniscono un aiuto HW con il **bit usata (o di riferimento)**
  - Ad ogni pagina è associato un bit, inizialmente = 0
  - Quando la pagina è referenziata il bit è impostato a 1 dall'architettura di sistema
  - **Rimpiazzare la pagina che è a 0** (se ne esiste una)
  - **Non se ne conosce l'ordine d'uso tuttavia** (se ne esiste più di una)
    - Info che differenzia i vari algoritmi che implementano la politica LRU

Approssimazione dell'algoritmo LRU (2)

- **Algoritmo seconda possibilità (o dell'orologio)**
  - Ogni pagina ha **un bit usata** nella PT
  - Le pagine sono disposte in **una lista circolare**
  - Quando occorre selezionare una pagina vittima comincio **una scansione lungo la lista**:
    - se una pagina ha il bit di riferimento a 1, lo pongo a zero e passo alla successiva (la pagina rimane in memoria);
    - altrimenti la seleziono per sostituirla



Algoritmo della seconda possibilità (orologio) per la sostituzione della pagina



Approssimazione dell'algoritmo LRU

- **Algoritmo seconda possibilità migliorato o algoritmo NRU (*Not Recently Used*)**
  - **Si basa sui bit *usata e modificata*** come coppia ordinata
    - Il bit usata viene azzerato periodicamente e quindi ci da un indicazione sull'uso "recente"
  - Vengono eliminate nell'ordine (sempre circolare) pagine:
    1. (0,0) non usate e non modificate
    2. (0,1) non usate e modificate
    3. (1,0) usate e non modificate
    4. (1,1) usate e modificate

Altri algoritmi: algoritmi di conteggio

IDEA: Tenere un contatore del numero di riferimenti che sono stati fatti ad ogni pagina

- Algoritmo LFU (Least Frequently Used): sostituisce la pagina con il più basso conteggio
  - **Problema**: pagine usate massicciamente (conteggio elevato) in fase di inizializzazione e non più usate
- Algoritmo MFU (Most Frequently Used): sostituisce la pagina con il più alto conteggio
  - basato sul fatto che la pagina con il conteggio più basso è stata probabilmente appena portata dentro e deve ancora essere usata

Poco usati (implementazione costosa) e non approssimano bene l'algoritmo utopia

Allocazione dei frame

Come e quanta memoria fissa allocare ai vari processi?

- **Ogni processo ha bisogno di un numero minimo di frame**
  - stabilito dall'architettura del computer (*instruction-set*)

- **Esempio: IBM 370** – 6 pagine per gestire l'istruzione MVC per muovere caratteri da memoria a memoria:
  - Istruzione a 6 byte, può occupare 2 pagine
  - 2 pagine per gestire il blocco di caratteri "da" muovere
  - 2 pagine per gestire la zona "verso" cui muovere

Algoritmi di allocazione dei frame

- Due principali schemi di allocazione:
  - **Allocazione fissa**
  - **Allocazione a priorità**

Allocazione fissa

- **Allocazione omogenea o uniforme** – per esempio, se 100 frame e 5 processi, ognuno prende 20 pagine
- **Allocazione proporzionale** – si assegna la memoria disponibile ad ogni processo in base alle dimensioni di quest'ultimo

s_i = dimensione del processo p_i

$S = \sum s_i$

m = # totale dei frame

a_i = spazio allocato a $p_i = \frac{s_i}{S} \times m$

$m = 64$

$s_1 = 10$

$s_2 = 127$

$a_1 = \frac{10}{137} \times 64 \approx 5$

$a_2 = \frac{127}{137} \times 64 \approx 59$

Allocazione a priorità

- Se si vuole dare ai processi a più alta priorità più memoria (indipendentemente dalla loro dimensione)
- Usare uno schema di allocazione **proporzionale che**
  - **usa le priorità piuttosto che la dimensione**
  - **o una combinazione dei due**

Allocazione globale e locale

Un altro fattore nel modo in cui i frame sono allocati:

- **Sostituzione locale** – ogni processo effettua la scelta solo nel proprio insieme di frame allocati

- **Sostituzione globale** – permette ad un processo di selezionare un frame di sostituzione a partire dall'insieme di tutti i frame
  - anche se quel frame è correntemente allocato a qualche altro processo (che possibilmente ha un numero di priorità più basso)
  - un processo può prendere un frame da un altro

- **La sostituzione gloable è la più usata**, perchè da un miglior rendimento del sistema

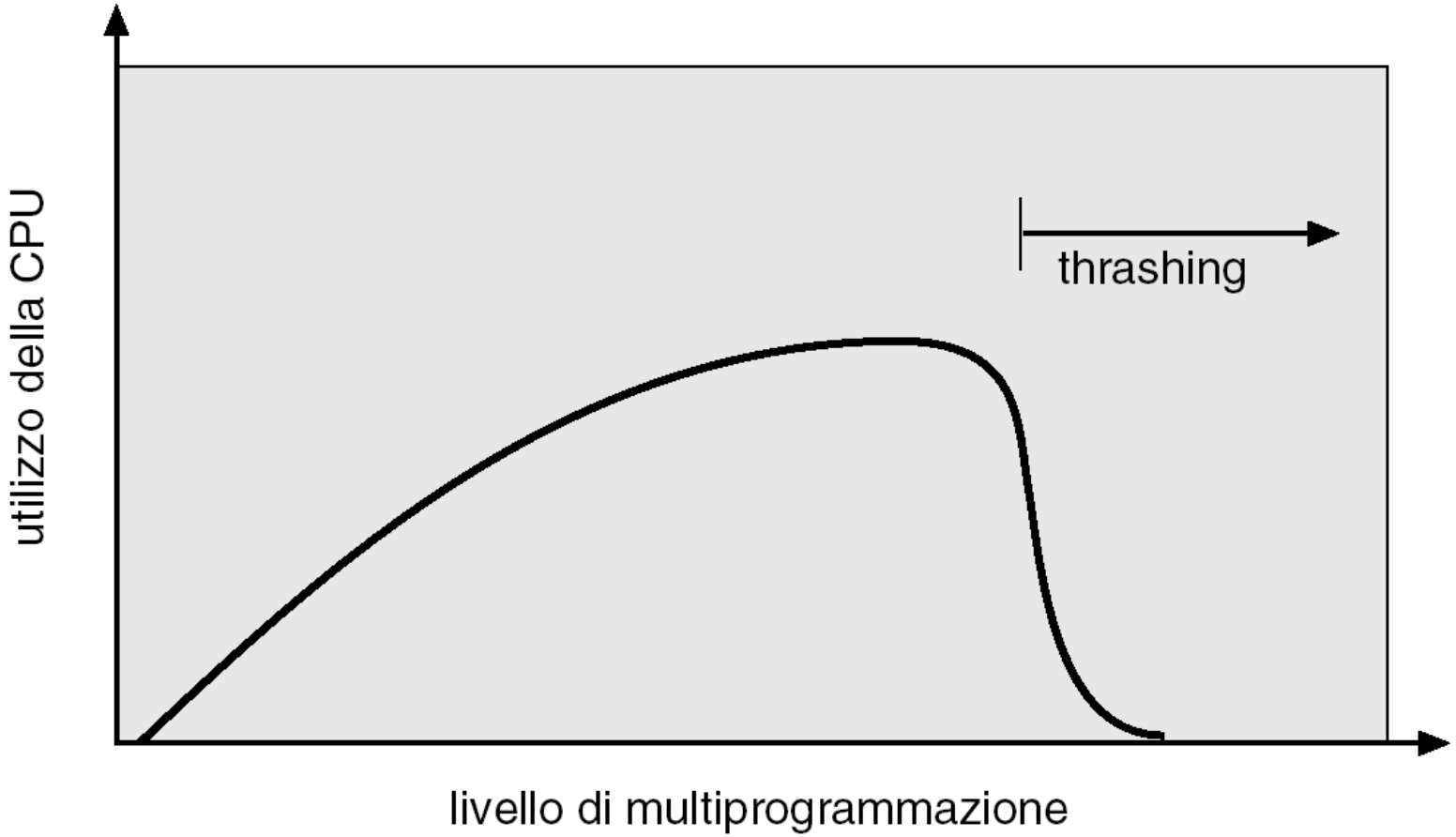
Thrashing (paginazione spazzatura)

Thrashing ≡ un processo spende **più tempo nella paginazione che** nella propria esecuzione

Cause del thrashing (presenti nei primi sistemi di paginazione):

- Se un processo non ha abbastanza pagine, il tasso di mancanza di pagina è molto alto
- il processo (in uno schema di allocazione globale) prenderà i frame degli altri processi e similmente questi
- Questo comporta:
  - la coda dei processi pronti si svuota immediatamente
    - i processi si mettono in coda nell'HW di paginazione
  - basso utilizzo della CPU
  - il SO ritiene che sia necessario aumentare il livello di multiprogrammazione
  - un altro processo aggiunto al sistema con calo di prestazioni

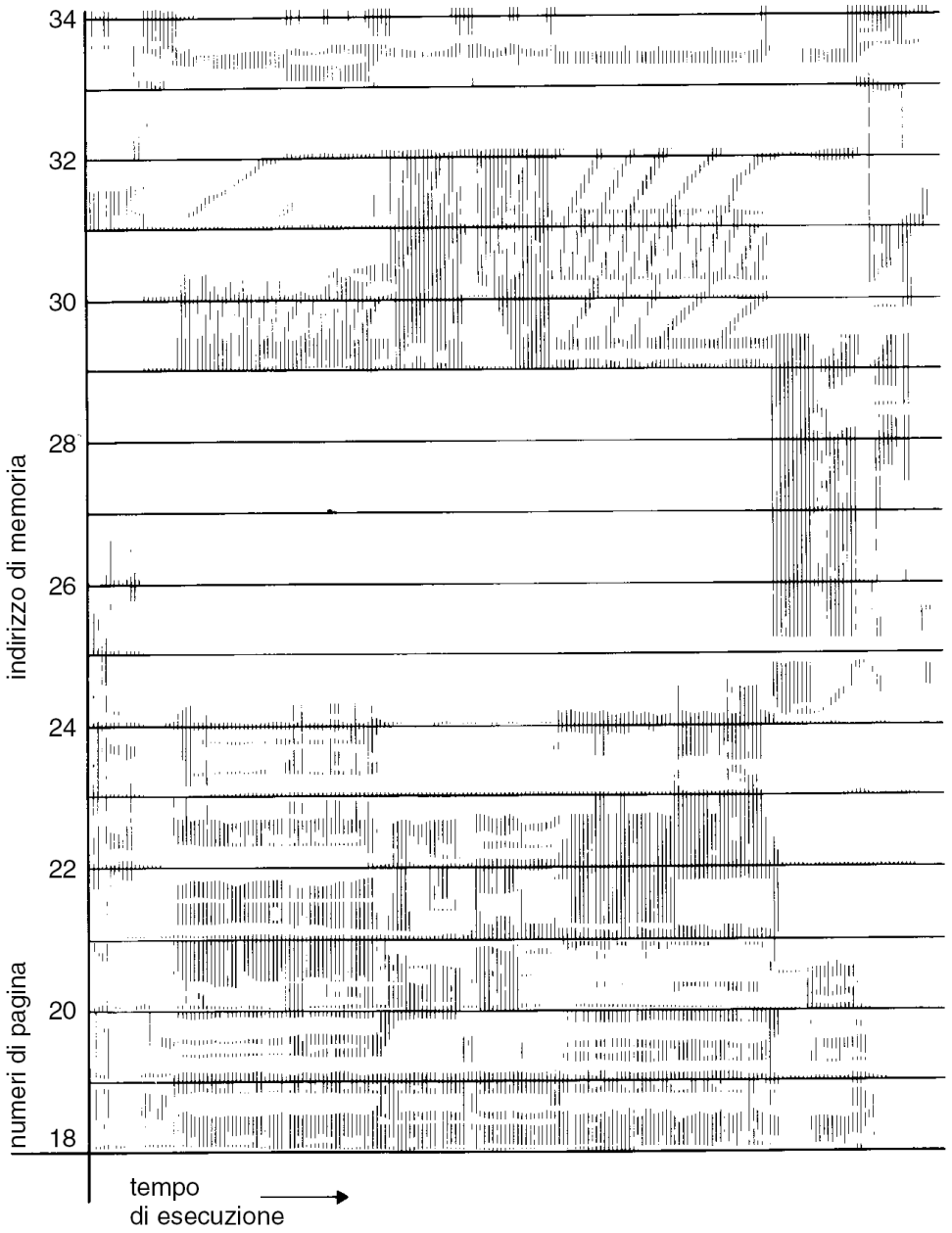
Thrashing e il livello di multi-programmazione



Limitare il Thrashing

- Occorre fornire ad un processo tanti frame quanti ne ha bisogno; ma "quanti"?
- Esistono diverse tecniche
- La strategia del "working set" inizia osservando quanti frame sta attualmente usando un processo
  - Basata sul **modello di località**:
    - Un programma è generalmente composto da varie località (insieme di *pagine attive*) che possono sovrapporsi
    - Un processo si muove da una località all'altra
    - **Perchè si verifica il trashing?**
      Σ dimensione della località > dimensione totale della memoria fisica allocata

Località in un modello di riferimento alla memoria



Blocco dei dispositivi di I/O

- Con la richiesta di paginazione, è utile permettere che alcune **pagine** siano **bloccate in memoria**
  - si associa un **bit di blocco** **ad ogni frame**, se bloccato il frame non può essere selezionato per la sostituzione
- **Usate in varie situazioni, ad es. per bloccare le pagine usate da "buffer" in operazioni di I/O**
  - Operazioni di I/O gestite direttamente da controllori di periferiche (ad es. per il disco) per conto di un processo sospeso in attesa del completamento (segnalato tramite interrupt)
  - Se non bloccate, le pagine potrebbero essere allocate ad altri processi
  - Una soluzione alternativa consiste nell'eseguire l'I/O nello spazio kernel (e non nello spazio utente)

Elaborazione in tempo reale

- La memoria virtuale garantisce un buon utilizzo del computer ottimizzando l'uso della memoria
- Tuttavia, i singoli processi possono soffrirne perché possono ricevere mancanze di pagine supplementari in esecuzione
  - Ritardi inaspettati
- Pertanto, la maggior parte dei **sistemi in tempo reale (hard) ed embedded non implementano la memoria virtuale**
- **Esiste anche "una via di mezzo": ad es. Solaris**
  - Un processo può comunicare quali pagine sono per lui importanti
  - Ad utenti privilegiati è permesso il **blocco delle pagine in memoria**